Performance Prediction of a Network-Centric Warfare System*

Insub Shin and Alexander H. Levis
System Architectures Laboratory
C3I Center, MSN 4D2
George Mason Unversity
Fairfax, VA 22030, USA
E-mail: {ishin, alevis}@gmu.edu

Abstract

When a system consisting of sub-systems is used for a time critical mission, the delays associated with the network connecting these sub-systems may play a critical role in battle management. Consequently, the combined models must be able to represent the network delay properly. In this paper, the architecture of a system is layered into two levels: a functional layer and a physical layer. Both architectural layers are developed as executable models: the functional executable model in a Colored Petri net and the physical executable model in a queueing net. Both layered executable models are synthesized to develop a performance prediction model. The message-passing pattern is generated from the Petri net using a state space analysis technique. Then, the queueing net model processes these messages preserving the message-passing pattern. Once the network delays are measured, the delay values are inserted into the Petri net model. The example in this paper shows how a small network delay in a C3 system affects the outcome of a time critical mission. It also illustrates design choices and how to develop tactics to resolve the tolerance of the network delays.

1 Problem Definition in a Network-Centric Warfare System

Many of the Command, Control, and Communication (C3) systems in the real world are supported by a common network or a network of networks. The concepts of "speed of command" and "self-synchronization" based on the Network-Centric Warfare formulation [Cebrowski and Garstka, 1997; Ceruti, 1998; Stein, 1998] introduce a new command and control

* This work was supported in part by the Air Force Office of Scientific Research under Grant No. F49620-98-1-0179

Report Documentation Page

Form Approved
OMB No. 0704-0188

1. REPORT DATE	2. REPORT TYPE	3. DATES COVERED
2000		**00-00-2000 to 00-00-2000**

4. TITLE AND SUBTITLE	5a. CONTRACT NUMBER
Performance Prediction of a Network-Centric Warfare System	5b. GRANT NUMBER
	5c. PROGRAM ELEMENT NUMBER
6. AUTHOR(S)	5d. PROJECT NUMBER
	5e. TASK NUMBER
	5f. WORK UNIT NUMBER
7. PERFORMING ORGANIZATION NAME(S) AND ADDRESS(ES) **George Mason University,System Architectures Laboratory,C3I Center,Fairfax,VA,22030**	8. PERFORMING ORGANIZATION REPORT NUMBER
9. SPONSORING/MONITORING AGENCY NAME(S) AND ADDRESS(ES)	10. SPONSOR/MONITOR'S ACRONYM(S)
	11. SPONSOR/MONITOR'S REPORT NUMBER(S)

12. DISTRIBUTION/AVAILABILITY STATEMENT
Approved for public release; distribution unlimited

13. SUPPLEMENTARY NOTES
The original document contains color images.

14. ABSTRACT

15. SUBJECT TERMS

16. SECURITY CLASSIFICATION OF:			17. LIMITATION OF ABSTRACT	18. NUMBER OF PAGES	19a. NAME OF RESPONSIBLE PERSON
a. REPORT **unclassified**	b. ABSTRACT **unclassified**	c. THIS PAGE **unclassified**		**14**	

(C2) requirements characterization. There are intensive information flows between sensors, C2, and shooters on a network of networks, and synchronization of information arrivals and updates. The availability of information at a certain time as well as the quality of information is critical. The elapsed time of each interacting component and the message delay can play a critical role in such a system of systems. The estimation of network delay is essential to verify if the system meets the timing constraints. Furthermore, the message delays from each component system can change the order of task execution and may cause a synchronization problem.

Modern surveillance and tracking systems often utilize multiple sensors of different types to provide complementary information. Multiple sensors or intelligence agents are connected through communications networks. The sensors are usually physically dispersed at geographic locations. Does the combination of the best sensors and the best communication systems guarantee the best outcome of combat results? The objective of this paper is to present methods to synthesize multiple architectural views and to estimate the connection delays between tasks so that the performance of a C3 system that performs multiple tasks in a real-time distributed computing environment can be predicted.

2 Synthetic Executable Model for Problem Solving

A C3 system for a high level military organization is a system-of-systems consisting of heterogeneous sub-systems operating under distributed system environments. The properties and behavior of a complex system are specified and expressed in multiple perspectives. In this paper, the logical behavior of a C3 system is specified in the functional architecture and the supporting resources are specified in the physical architecture. A functional architecture is a set of activities or functions arranged in a specified (partial) order that, when activated, achieves a defined goal. A physical architecture is a set of physical resources (nodes) that constitute the system and their connecting links.

Predicting the performance of a real-time distributed system requires synthesizing the multiple views together with the communication network. The traditional method is to represent the communication demand of each task and the contention of the demands as an embedded link in a

single integrated model. For example, $Link_{(k, k+1)}$ in Figure 1 is embedded in a single system. This approach may have limitations to represent the demand and the contention of network resources that are shared among multiple processes in a large-scale real-time distributed system.

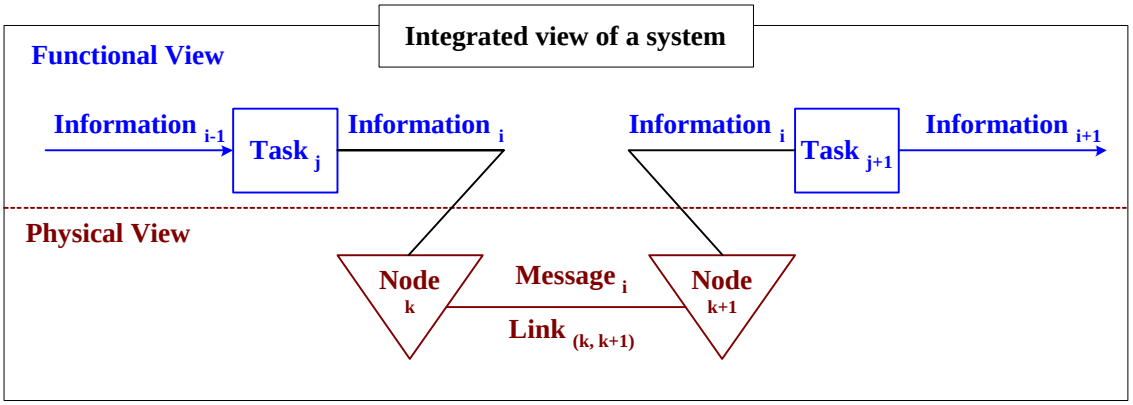

Figure 1. Integrated view of a system

Developing an executable model is a way for performance prediction. Operational formalisms such as Petri nets and state machines are suitable to specify and express executable models. Suppose that the two models are developed as shown in Figure 2 in which the functional architecture model is a Petri net and the physical architecture model is a queueing net.

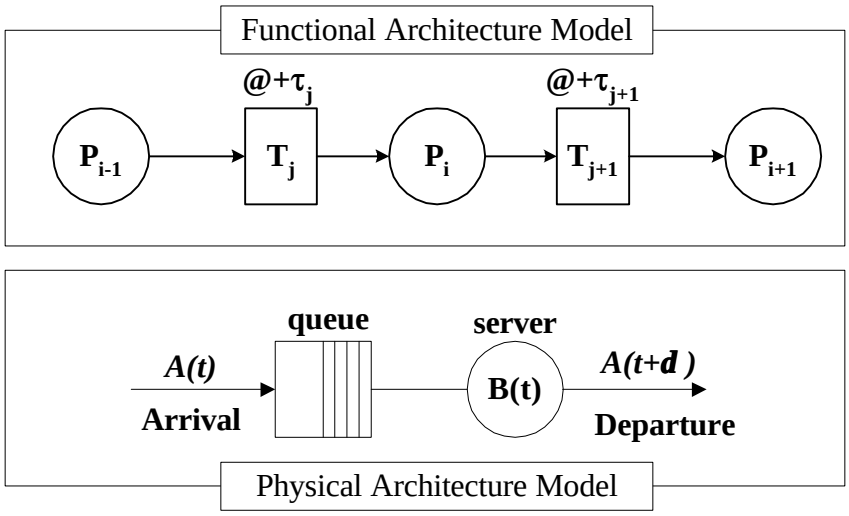

Figure 2. Two Separate Functional and Physical Architecture Models

The circle in the functional architecture model is a "place" and the bar is a "transition. Transition T_j has a processing delay τ_j other than network connection delays. The queueing net consists of a server and a queue depicted by a bar and a circle respectively. The server has a message processing rate B(*t*) at time *t*. If a message arrives at time t, depending the processing rate, the message will be delivered after a delay δ.

Central to synthesizing the two functional and physical executable architectures is the concept of two state machines communicating with each other. Once those executable models are developed, both models may communicate with each other during "run-time". Figure 3 depicts the run-time communication scheme between the two models. Once the functional executable model has a message at place P_{i-1} at time *t*, transition T_j will produce a message at place P_i at time $(t + \tau_j)$ after completion of the task. At the same time, this message arrives at the physical executable model. The physical executable model processes the message and produces the message after delay δ_j at time $(t + \tau_j + \delta_j)$. The message produced at place P_j after the firing of transition T_j in the functional architecture model is available at time $(t + \tau_j + \delta_j)$ when transition T_{j+1} can use the message.

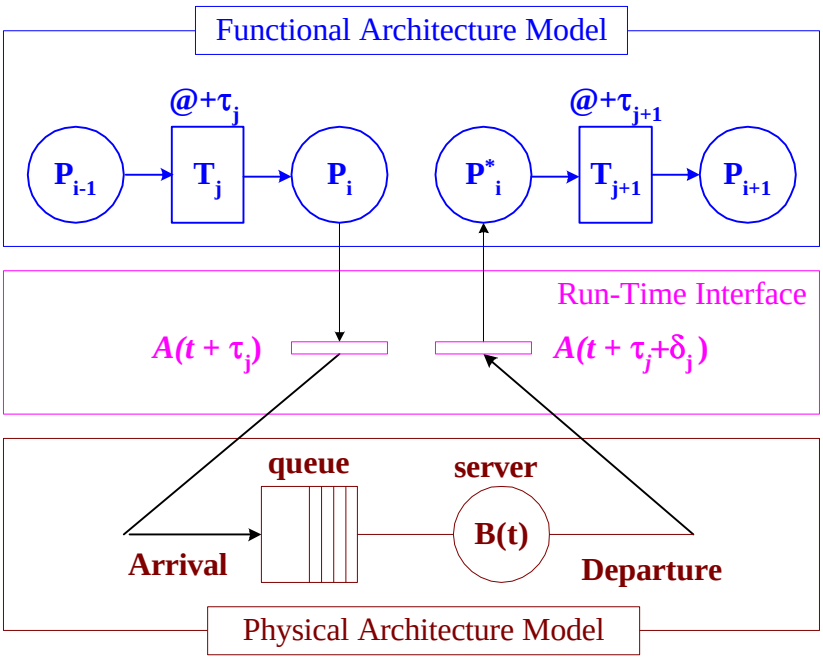

Figure 3. Run-Time Communication between Functional and Physical Architecture Models.

Instead of a run-time simulation of the two models, an “off-line” simulation technique is used in this paper. Figure 4 depicts the “off-line” communication scheme between the two models. The functional executable model sends its whole expected state transition information into the physical executable model to obtain the connection delays between tasks. Then, the physical executable model uses this state information for its event scheduling for simulation and produces network delay values corresponding to each logical connection link between tasks. Finally these delay values are inserted into the functional executable model. The resulting functional executable model with the estimated delay values is used for predicting the performance of the system.

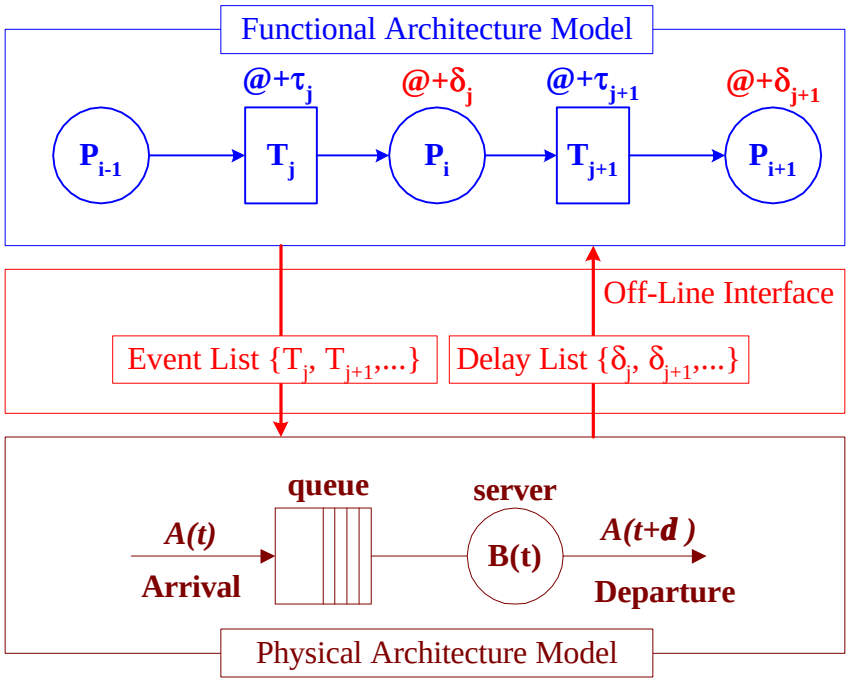

Figure 4. Off-Line Communication between Functional and Physical Architecture Models.

Figure 5 is an example of a specification of both functional and physical architectures and their relationships. The functional architecture in the example has three tasks (or functions); F_i, F_j, and F_k. The physical architecture provides services to carry out the tasks in the functional architecture. Resources R_i, R_j, and R_k can be interpreted as servers carrying out the tasks F_i, F_j, and F_k, respectively. The information flow between tasks in the functional architecture: ‘Info (i,

j)', 'Info (i, k)', 'Info (j, k)', 'Info (j, i)', and 'Info (k, j)' generate message-passing between resources in the physical architecture. In turn, this message passing generates traffic over the networks between source-destination node pairs (N_i, N_j), (N_i, N_k), (N_j, N_k), and (N_j, N_i), respectively. Using this mechanism, if we know the information flow pattern in the functional architecture, we can represent the realistic physical traffic flow over the communication network through the series of logical tasks executions.

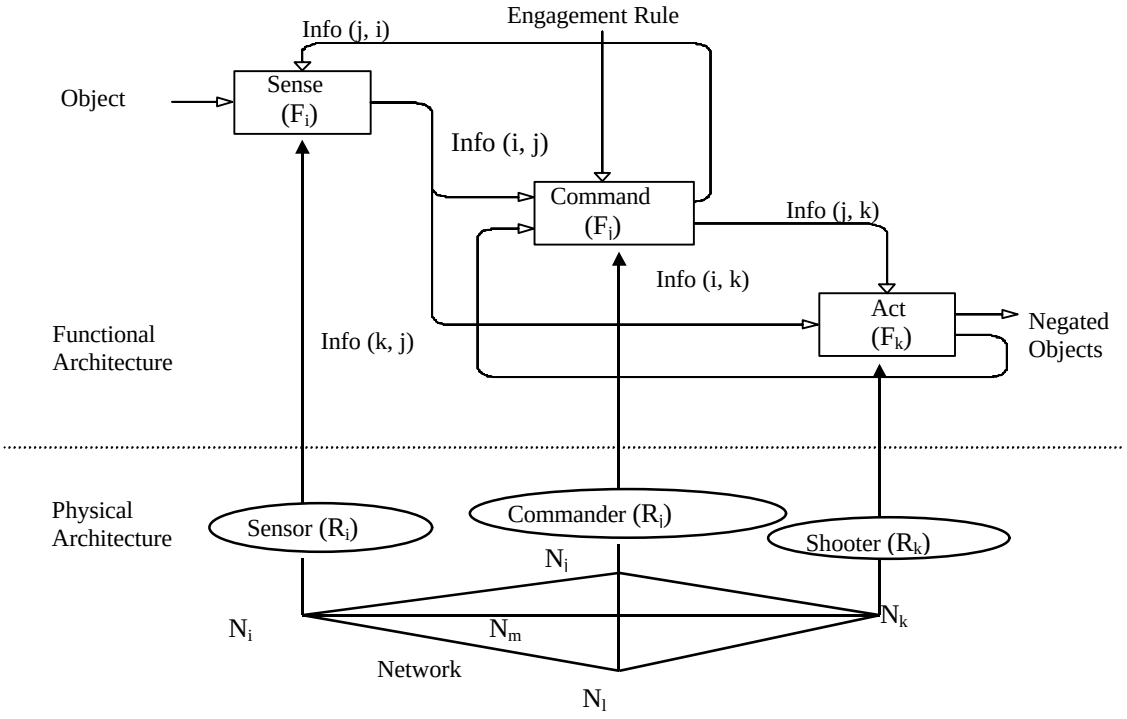

Figure 5. Example of Architectures

3 Anti-Air Warfare System: an Example

3.1 Scenario

Suppose that an Anti-Air Warfare (AAW) system carries out a mission of negating an incoming threat as shown in Figure 6. The mission is to negate air threats when multiple unknown objects are approaching friendly forces with different velocities at different locations.

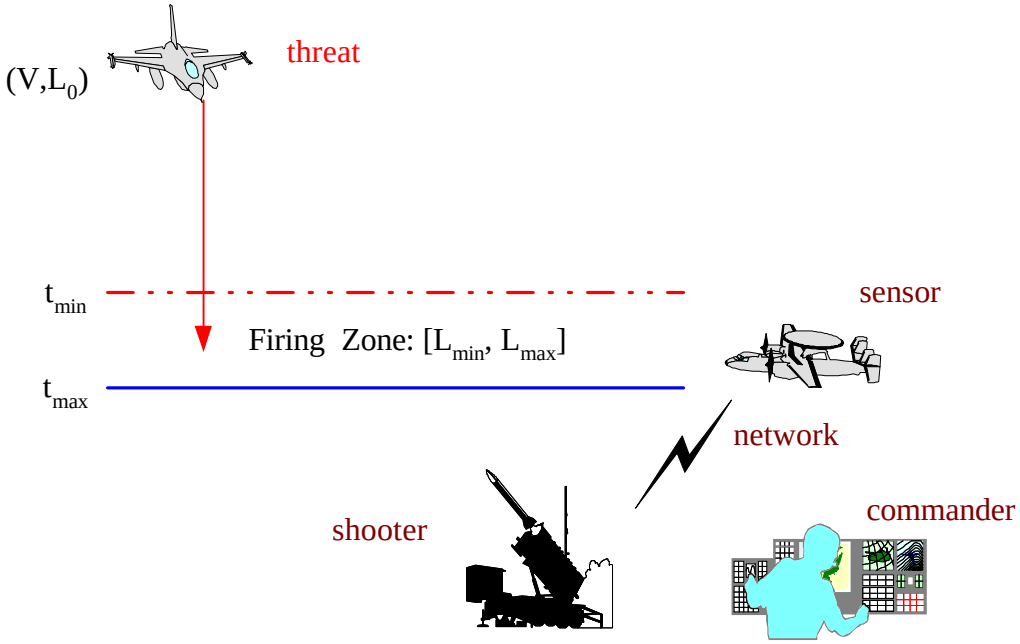

Figure 6. Operational Concept Diagram of the AAW System

The scenario is follows. If the flight is hostile, it must be intercepted before it arrives at the frontal battle line or when it moves into a specified area (firing zone), resulting in a timing constraint. Neutral flights must not be killed. To carry out the mission, the AAW system is composed of three tasks; *sense*, *command*, and *act*. Each task is carried out by assigned assets; *sensor*, *commander*, and *shooter*, respectively, and the assets are connected on a network. Each asset has its own information processing devices to carry out the assigned task. An early warning system provides information for the identification of the objects with a Bayesian fusion algorithm. A tracking radar provides the estimated location of flying targets with a Kalman filter algorithm. One air defense unit has one tracking radar with capability to track multiple targets, one surveillance radar to identify the targets, and one missile launcher with the capability to arm multiple missiles. An early warning system that is geographically distributed over the battlefield supports the air defense unit for identification and tracking of targets

From the mission needs, we can derive the timing constraints of the AAW system as follows. The system must respond against the threat when the threat is located within the interval [L_{min},

L_{max}]. Let V be the velocity of the flight and L_0 be the initial location of the flight. Then the system must respond to the external stimuli within the time interval [t_{min}, t_{max}], where

$$t_{\min} = (L_{\min} - L_0)/V \qquad \text{and} \qquad t_{\max} = (L_{\max} - L_0)/V$$

This is the timing constraint on the AAW system. The AAW system must respond to the threat after t_{min} but before t_{max} and the response must match the desired goal of the mission. The accuracy of the response is another constraint on the AAW system. In the example scenario, the mission needs statement "the threat must be negated if the flight is hostile" specifies the required accuracy of the response.

The threat is characterized by the identity (e.g., hostile flight, friendly force flight, decoy, and a flock of birds, etc.), the initial location, the velocity, and the number of threats. Each threat has a different probability of detection according to the avenue of approach such that the probability is low in mountainous areas and high in open areas. The interceptor of the AAW system is characterized by the location of initial deployment, the velocity, the firing range, and the probability of kill. The AAW should be able to handle multiple threats. But there are limited resources. Also one launcher can load multiple missiles at the same time and it takes a certain time to reload after firing one.

The output of the combat operations is assessed as one of three combat results; 'kill enemy (H)', 'kill neutral (H^c)', and 'leak'. The results 'Kill' or 'Leak' is affected by the response time of the system and 'kill H' or 'kill $H^{c'}$ is affected by the uncertainty level of the target at the decision time. Whether the interceptor 'kills' or 'misses' the target depends on the probability of kill. The probability depends on the accuracy of the estimated location of the target. If the variance of the accuracy is within the hit range of the interceptor, the target is killed. Once the result is identified as a 'miss,' the interceptor fires repeatedly while the target is within the window of opportunity.

The sensor is characterized by the types (i.e., identification and tracking), the sensitivity (i.e., likelihood ratio of positive detection over false detection) of identification, the variance of

estimated location and velocity, the surveillance range, scan cycles, and the number of sensors. Multiple sensors are geographically dispersed and configured for centralized data fusion.

The problem in question is to choose a design option to carry out the mission successfully.

3.2 Experimental System Design

The system is expected to be used under different tactical operational environments. To represent the situation, two rules of intercept have been established:

- *Aggressive Intercept*: when the object is identified as a threat (i.e., a hostile target), the system should intercept it regardless of the current location of the target before the object arrives at the frontal battle line.
- *Defensive Intercept*: when the object is identified as a threat (i.e., a hostile target), the system should intercept it when the target is within a specified firing zone.

The effectiveness of the system was measured by:

- *Mission Success*: the ratio of the number of threats cleared over the total number of threats within the allowed time; it measures performance in a [0.0, 1.0] scale.
- *Decision Quality*: the ratio of the number of correct decisions over the total number of decisions; it measures performance in a [0.0, 1.0] scale.
- *Weapon Efficiency*: the ratio of the number of threats cleared over the number of interceptors (i.e., missiles) used; it measures performance in a [0.0, 1.0] scale.
- *Operational Readiness*: the time at which the system finishes all the tasks and switches the operational mode into the idle state; it measures performance in units of time.

An executable performance prediction model has been designed with seven input parameters in addition to the rules of intercept and four output performance measures. The parameters are:

- *Number of Sensors:* the number of surveillance and tracking radar
- *Number of Transponders*: the number of transponders per each radar.

- *Sensor Likelihood Ratio*: probability of positive identification over probability of false identification.
- *Computation Rule for Fusion*: *{dynamic, fixed}*.When some sensor data is unavailable due to any reason (a sensor is out-of-service, communication links are jammed, a target is not detected, or sensors in operation have different surveillance ranges, etc.), the data fusion computation needs to have a rule to handle the situation, otherwise it will have to wait until the remaining sensor data arrives; if more than one sensor is in operation and if the sensor data is received from only a subset of sensors at a specific time, the 'dynamic' computation rule uses the available sensor data for fusion (a set of conditional probability matrices was used for this rule), while the 'fixed' computation rule stores the data from each sensor and uses the stored data together with the partial information received.
- *Surveillance Mode*: {*active, passive*}. If the uncertainty of a target does not meet the decision criterion, the 'active' rule assigns priority to using a specific sensor for the specific target until the uncertainty decreases to make a judgment; the 'passive' rule scans the battlefield routinely without the priority; when the target is a High Valued Target (HVT), the HVT will have priority so that the target is intensively searched and tracked. In the *passive* mode, the sensor scans the battlefield periodically.
- *Information Delay*: delay including information processing time and information distribution delay over the network.
- *Decision Threshold*: decision criterion to make a judgment whether the object is a hostile target or not; if the probability of the target identity is higher than the criterion, the target is considered as a hostile target and a fire order is issued.

This experimental system was implemented by Design/CPN [1999] for the functional executable model and Network Simulator [1999] for the physical executable model.

3.3 Findings

- *Number of Sensors and/or Transponders:* In general, more sensors or more transponders produce higher effectiveness. However, the relation was not true in some cases. In the *dynamic* computation rule for data fusion, as the number of sensors increases, the system

shows better timeliness (operational readiness). However, the opposite was true in the *fixed* computation rule.

- *Computation Rule:* The *dynamic* computation for data fusion supports better timeliness (operational readiness) than the fixed computation rule does, even though the decision criterion of the *dynamic* computation rule has a higher threshold than that of the *fixed* computation rule.
- *Number of Sensors vs. Computation Rule:* In the *dynamic* computation for data fusion, as the number of sensors increases, the system shows better timeliness (operational readiness). However, the opposite was true in the *fixed* computation rule.
- *Decision Threshold:* Clearly, there is a relationship between the decision criterion and the success of mission. If we focused on the timeliness of a response, the lower threshold for the decision criterion is preferred. However, if we focused on the accuracy of a response, the higher threshold is preferred.
- *Decision Threshold vs. Computation Rule:* In the *dynamic* computation rule for data fusion, the decision threshold needed to be set to the higher value for better performance. This is reasonable because this computation rule loses the benefit of multiple sensors since it uses a partial number of sensors, and so the higher threshold is required to get higher quality of information. Conversely, in the *fixed* computation rule for data fusion, the decision threshold needed to be set to the lower value. This rule fuses data assuming there are multiple sets of sensory data. It takes a long time to reach the higher threshold when some sensory data is unavailable. So it is reasonable that the lower threshold is required to respond quickly.
- *Information Delay:* Obviously the information delay should affect the timeliness of the response. But it was not true that the minimum information delay always guarantees the fastest response. It means that the information delay affects the behavior of the system not just in terms of the timeliness, but also the ordering of tasks in an autonomous system depending on the arrival time of information. It is the authors' conjecture that, once the information processing and distribution systems have the potential capability to carry out the mission successfully, the performance of the system (response time) may be more highly affected by the order of the information arrivals than by the delay of information.

Throughout the results, when a system consisting of a data fusion sub-system with a single transponder was used, the possible combinations of input parameters to meet the performance requirements were restricted. When multiple transponders were used, there were many more possible combinations of input parameters that met the performance requirements.

Based on this experiment, the *fixed* computation rule shows better *weapon efficiency* than the *dynamic* computation rule does. It can be interpreted that the estimation of the tracking system with the *fixed* computation rule provides the accurate target trajectory. This is reasonable for this experiment since the dynamic computation rule uses the partial set of local sensory data at the time of computation disregarding the old information stored at the central data fusion system while the fixed rule uses the full sensory data stored at the central data fusion system. For more realistic analysis, the tracking algorithm may need to be implemented in more detail.

3.4 Answers to War-Fighter's Questions

In general, the *number of transponders* per sensor has a critical impact: the more transponders the better. Given the number of transponders, if there are multiple sensors and if the data fusion system is used in an autonomous firing mode by the computerized automated system,

- under an unstable system operational environment due to any reasons (poor survivability of sensors, jamming on the communications link, etc.),
- if the ranges of surveillance radars are different, or
- if the probability of target detection is low

then, the data fusion system must be designed using the *dynamic* computation rule for data fusion and the decision criterion for the judgment of threat identification needs to be established with the higher threshold. More sensors will increase system performance.

If the data fusion system is designed using the *fixed* computation rule, as the number of sensors increases, there is possibility that the system effectiveness may decrease. So the number of sensors needs to be chosen properly. More experiments are required using parameters under the stable system operational environment, identical surveillance ranges, and higher probability of

target detection. In the scenario of this experiment, the lower decision criterion threshold increased performance. The decision threshold needs to be reexamined if the *fixed* computation rule is used under different scenarios.

4 Conclusion and Future Works

This paper presented how to synthesize a functional architecture and a physical architecture to predict the performance of a networked real time system. The case study model also illustrated how to develop tactics to resolve the tolerance of the network delays in a network centric warfare system.

Since the communication service demands are isolated from the functional model, the communications network can be specified in any preferred level of detail independently. This enables the logical model to be invariant with respect to the physical model resulting in flexibility in designing a large-scale C3 information system. The C4ISR Architecture Framework specifies three views: operational architecture view, systems architecture view, and technical architecture view. The proposed synthetic simulation technique and the invariant feature of the logical model allow collaborative work in developing those operational architecture and system architecture views.

Currently the simulation of the communication network schedules the traffic in a total order over a single time line. The future research is to expand the synthesis technique to deal with multiple operational architectures with non-monotonic simulation time.

5 References

[Cebrowski and Garstka, 1997] Arthur K. Cebrowski and John J. Garstka. "Network-Centric Warfare: Its Origin and Future," U.S. Naval Institute. [Online] Available at http://www.usni.org/Proceedings/Articles98/ PROcebrowski.htm, 1997.

[Ceruti, 1998] M.G. Ceruti "Challenges in Data Management for the United States Department of Defense (DoD) Command, Control, Communications, Computers, and Intelligence (C^4I) Systems," *Proceedings of the twenty-second Annual International Computer Software and*

Applications Conference, COMPSAC '98, 19-21 Aug., 1998, pp. 622-629.

[Design/CPN, 1999] Design/CPN Online, http://www.daimi.au.dk/designCPN/

[Network Simulator, 1999] Network Simulator Online, http://www-mash.cs.berkeley.edu/ns/

[Shin and Levis, 1999] Insub Shin and Alexander H. Levis, “Performance Prediction Model Generator Powered by Occurrence Graph Analyzer of DesignCPN”, *Proc. of 2nd Workshop on Practical Uses of Coloured Petri Nets and Design/CPN*, DAIMI PB-532, Aarhus University, Denmark, October, 1999.

[Stein, 1998] Fred Stein. "Observations on the Emergence of Network Centric Warfare." *Proceedings of the 1998 Command and Control Research & Technology Symposium*, June, 1998.